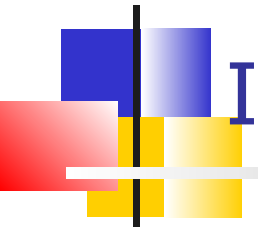
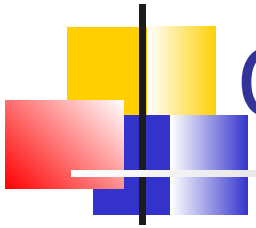


Introdução a UML e diagrama de classes

Turmas D e E

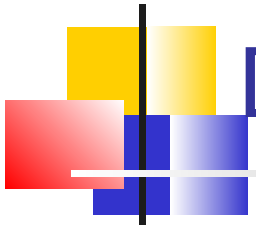
Juliana Félix (julianafelix@ufg.br)
Dirson S. Campos (dirson_campos@ufg.br)

21/02/2022



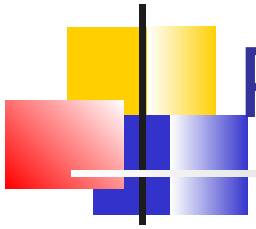
O que é UML ?

- A UML (*Unified Modeling Language*) é uma linguagem para **especificação**, **documentação**, **visualização** e **desenvolvimento** de sistemas orientados a objetos.
- Considerada uma das linguagens **mais expressivas** para modelagem de sistemas orientados a objetos.
- É possível representar sistemas de softwares sob **diversas perspectivas** de visualização.
- Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema:
  - Gerentes;
  - Coordenadores;
  - Analistas e
  - Desenvolvedores.



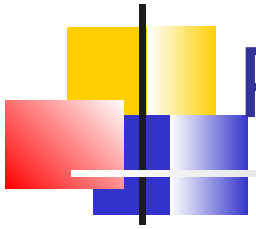
Desenvolvimento de Software

- Etapas
  - Levantamento de Requisitos
  - Análise
  - Projeto
  - Implementação
  - Testes
  - Implantação



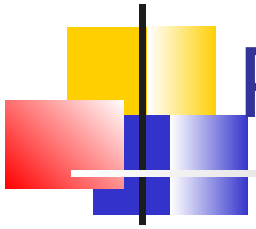
Principais Diagramas

- Categorias:
  - Diagramas Comportamentais
  - Diagramas Estruturais



Principais Diagramas

- Diagramas Comportamentais
  - Casos de Uso
  - Transição de Estados
  - Atividades
  - Diagramas de Interação
    - Sequência
    - Colaboração/Comunicação

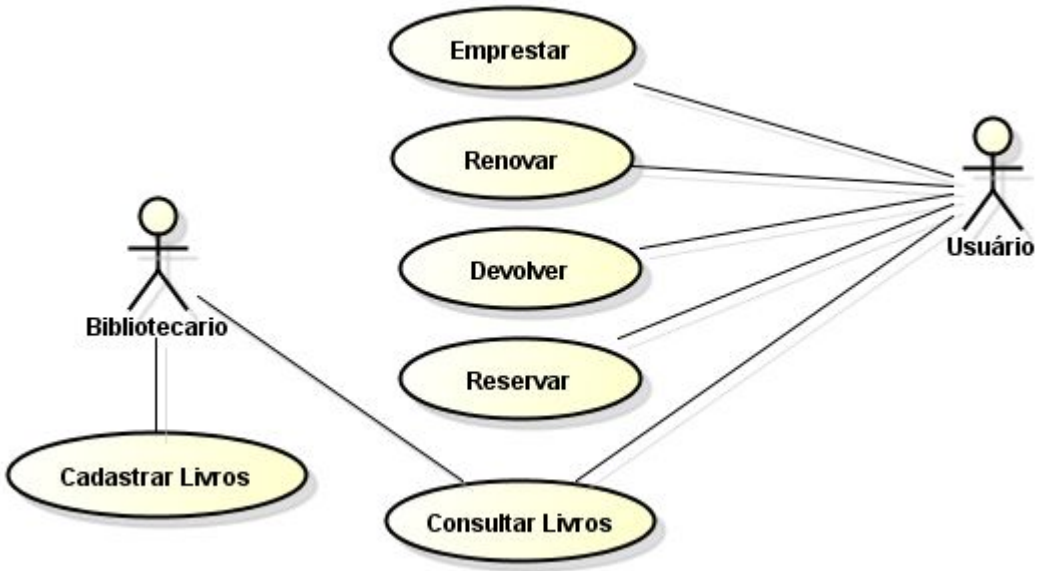


Principais Diagramas

- Diagramas Estruturais
  - **Classes**
  - Objetos
  - Componentes
  - Pacotes
  - Implantação/Instalação

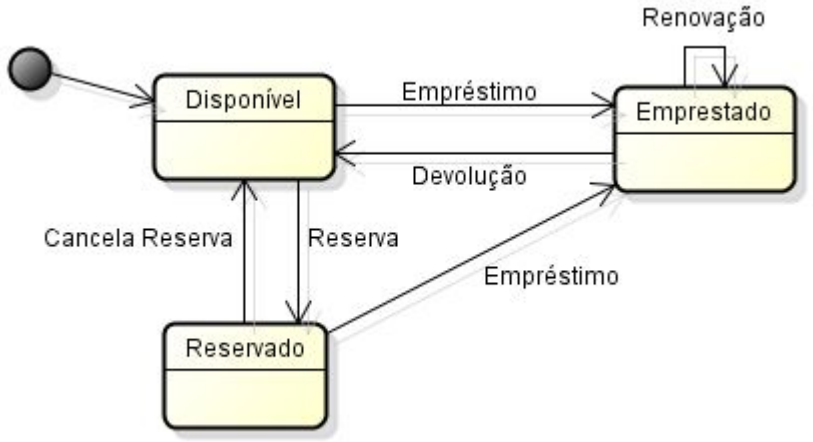
Principais Diagramas

- Casos de Uso
  - É um diagrama usado para identificar comportamentos diferentes do sistema. Representa os atores e suas operações.
  - Exemplo: Casos de uso de uma Biblioteca



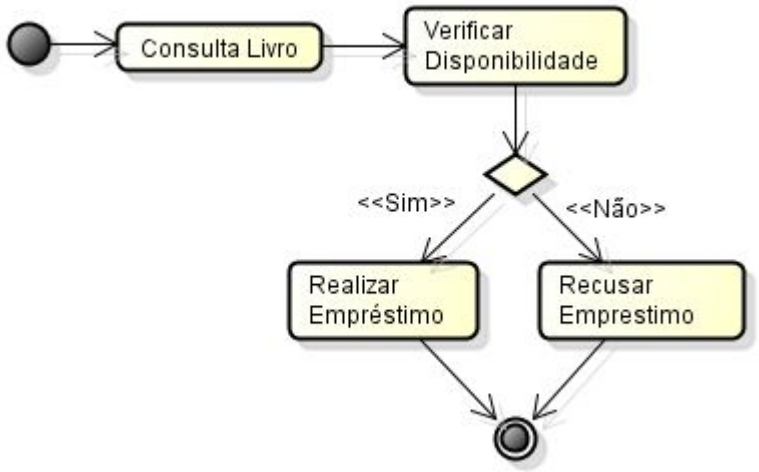
Principais Diagramas

- Transição de Estados
  - Representa um conjunto de estados que um objeto pode estar e os eventos que estimulam a transição de um estado para o outro.
  - Um Livro em uma biblioteca pode estar: Disponível, Emprestado, Reservado.



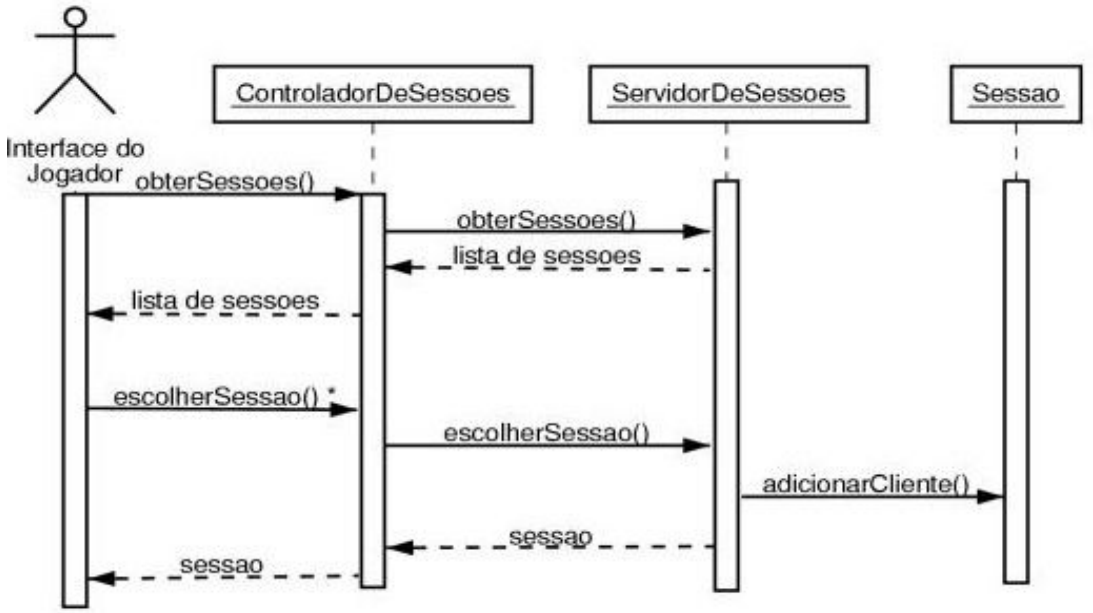
Principais Diagramas

- Diagrama de Atividades
  - O objetivo do diagrama de atividades é mostrar o fluxo de atividades em um único processo. O diagrama mostra como as atividades dependem uma das outras.
  - Exemplo: Emprestar Livro



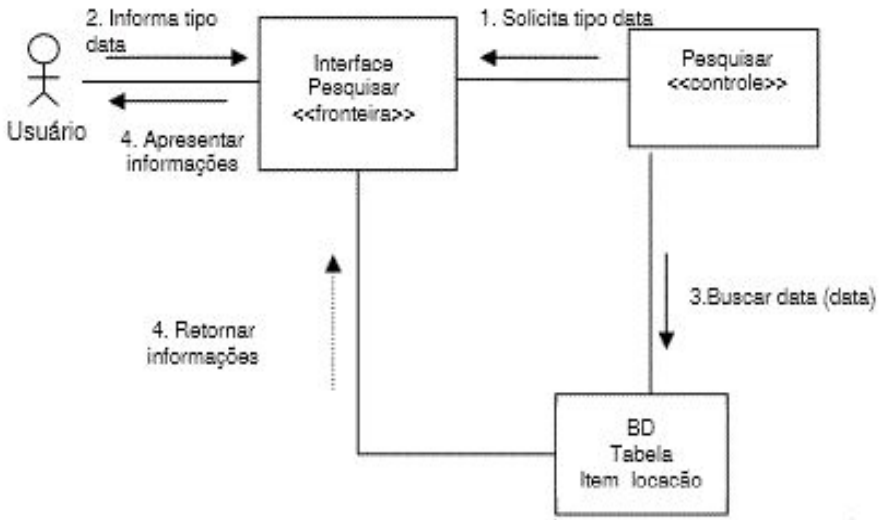
Principais Diagramas

- Diagrama de Sequência
  - Representa uma perspectiva orientada por tempo da troca de mensagens (chamada de métodos e retornos) entre os objetos.
  - Exemplo: Estabelecendo uma sessão



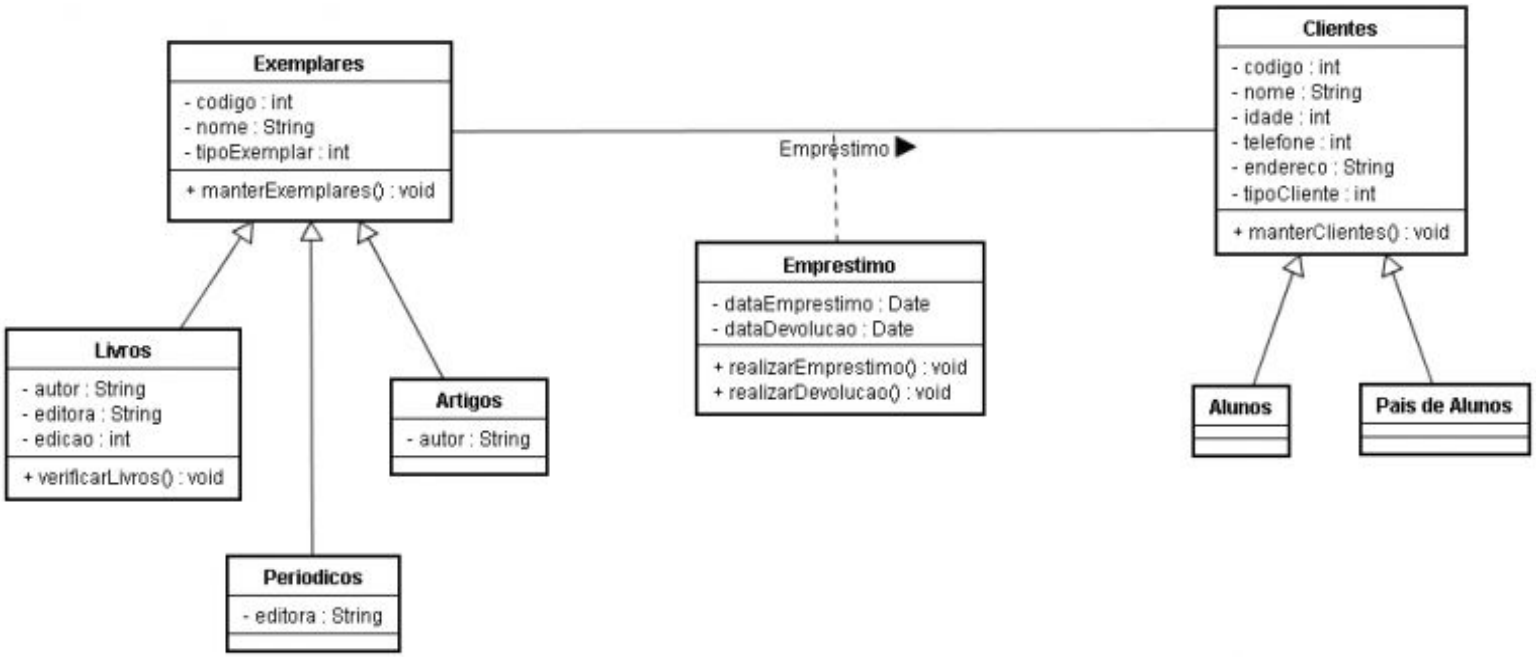
Principais Diagramas

- Diagrama de Colaboração/Comunicação
  - Representa um conjunto de objetos que colaboram para um comportamento do sistema – mostra a troca de mensagens com ênfase na ordem.
  - Exemplo: Pesquisar item de locação



Principais Diagramas

- Diagrama de Classes
  - Representa uma coleção de classes do sistema com seus relacionamentos.
  - Exemplo: Clientes com seus Empréstimos

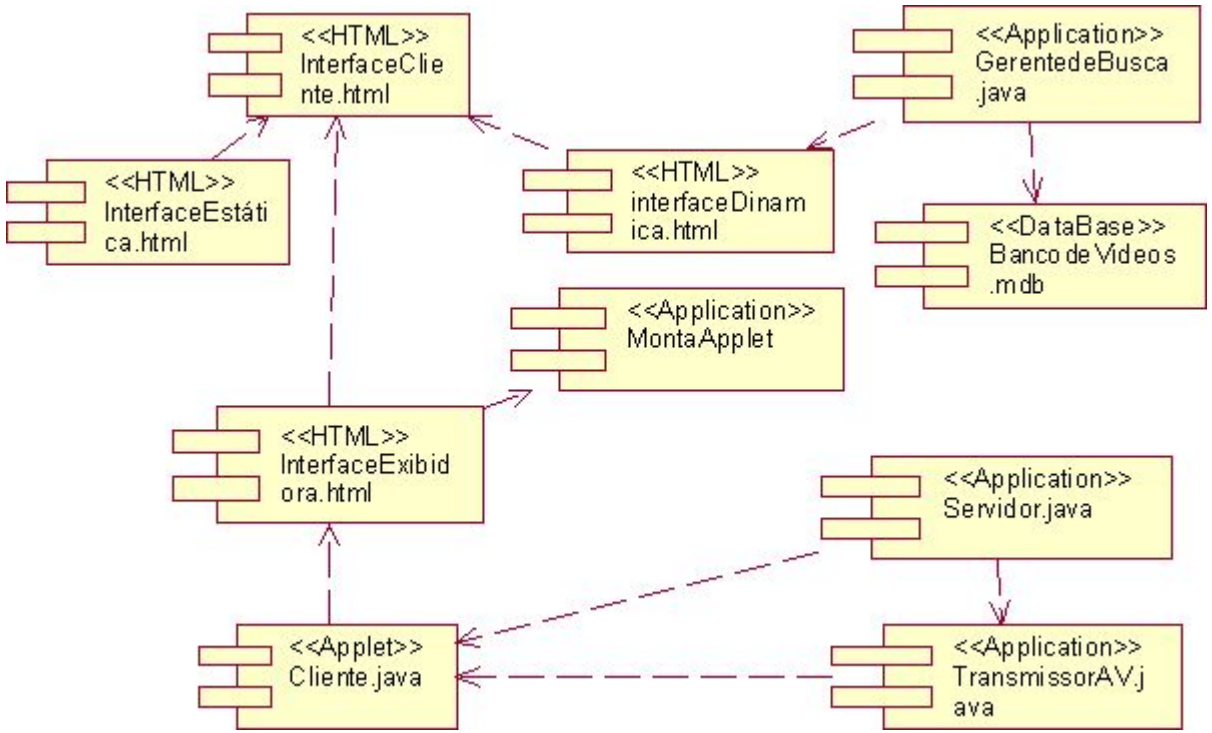


Principais Diagramas

- Diagrama de Objetos
  - Representa retrato em tempo de execução dos objetos com seus relacionamentos.
  - Exemplo: Clientes com contratos de aluguel de carro

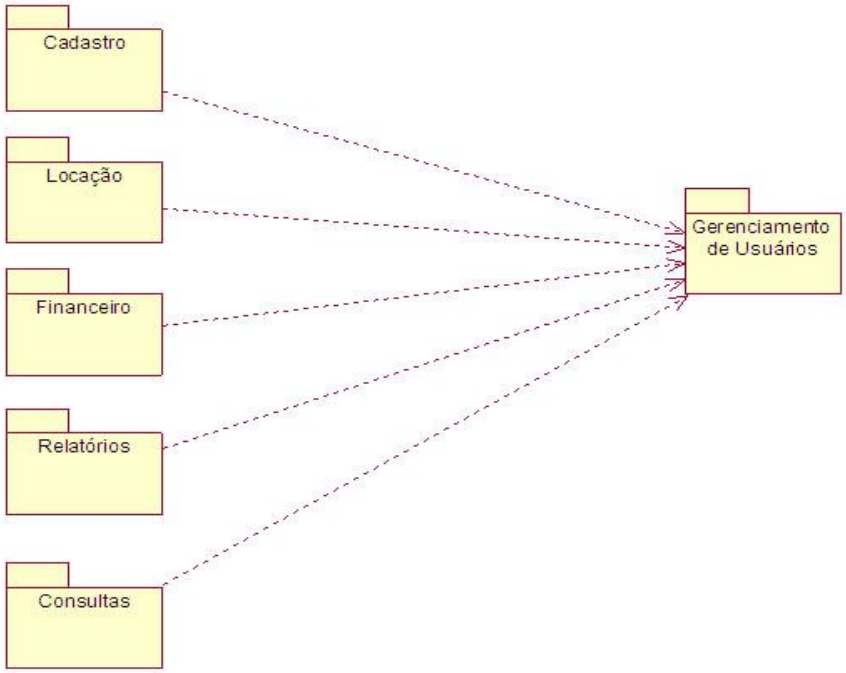
Principais Diagramas

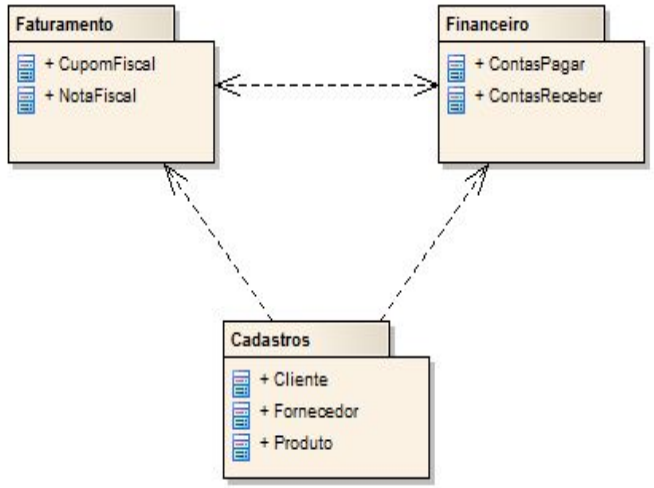
- Diagrama de Componentes
  - Representa um conjunto de componentes de software e seus relacionamentos.



Principais Diagramas

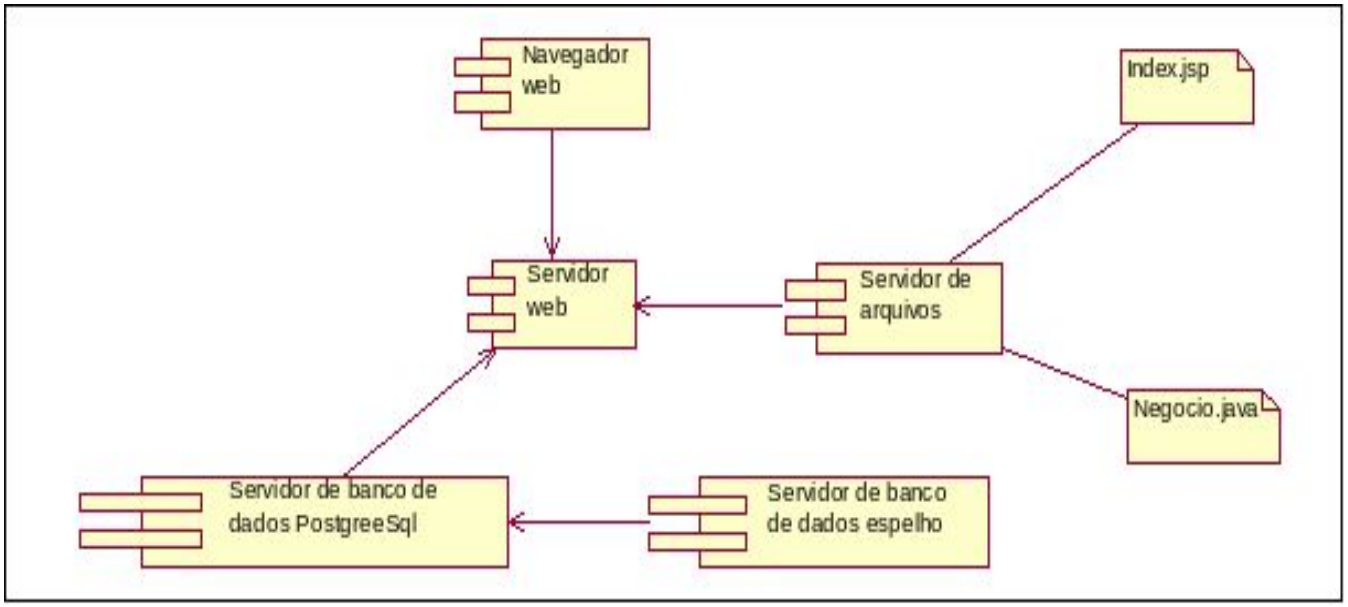
- Diagrama de Pacotes
  - Representa como os elementos do sistema estão agrupados





Principais Diagramas

- Diagrama de Implantação/Instalação
  - Representa a configuração e arquitetura do sistema através de componentes lógicos, físicos e suas interações.



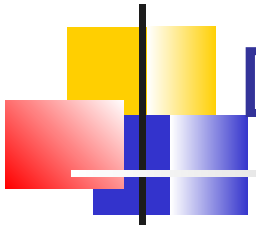


Diagrama de Casos de Uso

- O Diagrama de Casos de Uso tem o objetivo de auxiliar a comunicação entre os analistas e o cliente.
- Um diagrama de Caso de Uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.
- O cliente deve ver no diagrama de Casos de Uso as principais funcionalidades de seu sistema.

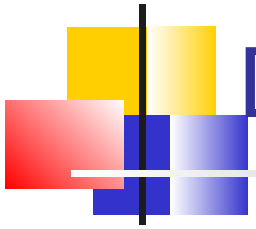


Diagrama de Casos de Uso

Notação

- O diagrama de Caso de Uso é representado por:
  - atores;
  - casos de uso;
  - relacionamentos entre estes elementos.

- casos de uso podem opcionalmente estar envolvidos por um retângulo que representa os limites do sistema.

Diagrama de Casos de Uso

Ator

- Um ator é representado por um boneco e um rótulo com o nome do ator.
- Um ator é um usuário do sistema, que pode ser um usuário humano ou um outro sistema computacional.

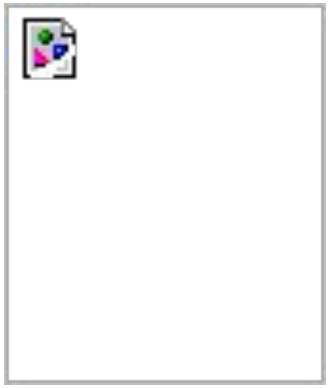


Diagrama de Casos de Uso

Caso de Uso

- Um caso de uso é representado por uma elipse e um rótulo com o nome do caso de uso.
- Um caso de uso define uma grande função do sistema.
- Como uma função pode ser estruturada em outras funções, um caso de uso também pode ser estruturado.

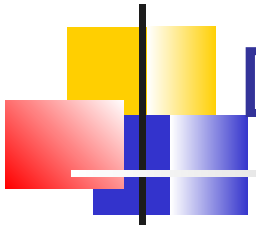


Diagrama de Casos de Uso

Relacionamentos

- Ajudam a descrever um caso de uso;
- associações entre atores e casos de uso;
- generalizações entre os atores;
- generalizações, *extends* e *includes* entre os casos de uso.

Diagrama de Casos de Uso

Relacionamentos

- Entre um ator e um caso de uso
  - Associação

Diagrama de Casos de Uso

Relacionamentos

- Entre um ator e um caso de uso - **Associação**
  - Define a interação de um ator com uma funcionalidade do sistema
  - Através desse relacionamento sabemos quem são os atores envolvidos em uma funcionalidade

Diagrama de Casos de Uso

Relacionamentos

- Entre atores - **Generalização**
  - Define que atores herdam casos de uso de outros atores, compartilham casos de uso
  - A compartilha os casos de uso de B
  - A tem seus próprios casos de uso

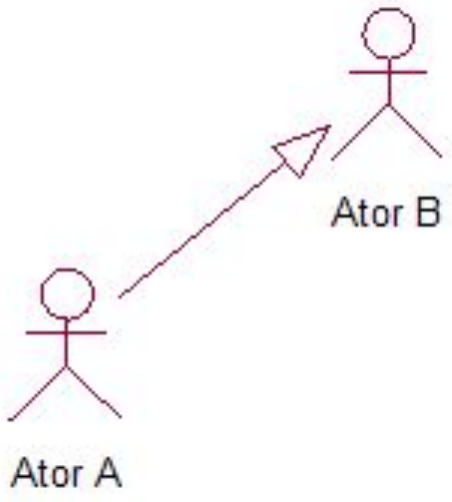


Diagrama de Casos de Uso

Relacionamentos

- Entre casos de uso – ***Include***
  - Um relacionamento ***Include*** de um caso de uso A para um caso de uso B indica que B é essencial para o comportamento de A.
  - Pode ser dito também que B é parte de A.

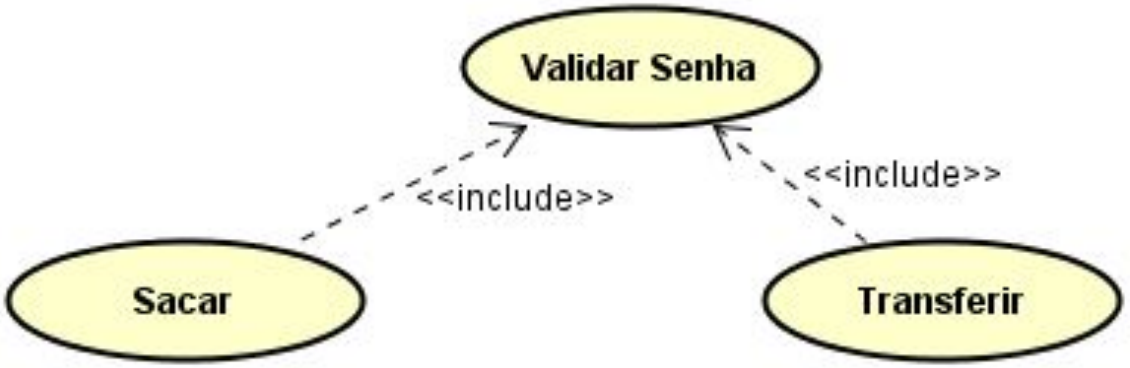


Diagrama de Casos de Uso

Relacionamentos

- Entre casos de uso – ***Extend***
  - Um relacionamento ***Extend*** de um caso de uso B para um caso de uso A indica que o caso de uso B pode ser acrescentado para descrever o comportamento de A (não é essencial)
  - B é uma variação de A. Contém eventos adicionais, para certas condições

Diagrama de Casos de Uso

Relacionamentos

- Entre casos de uso – ***Generalização ou Especialização (é_um)***
  - caso de uso B é_um caso de uso A (A é uma generalização de B, ou B é uma especialização de A).
  - Um relacionamento entre um caso de uso genérico para um mais específico, que herda todas as características de seu pai.

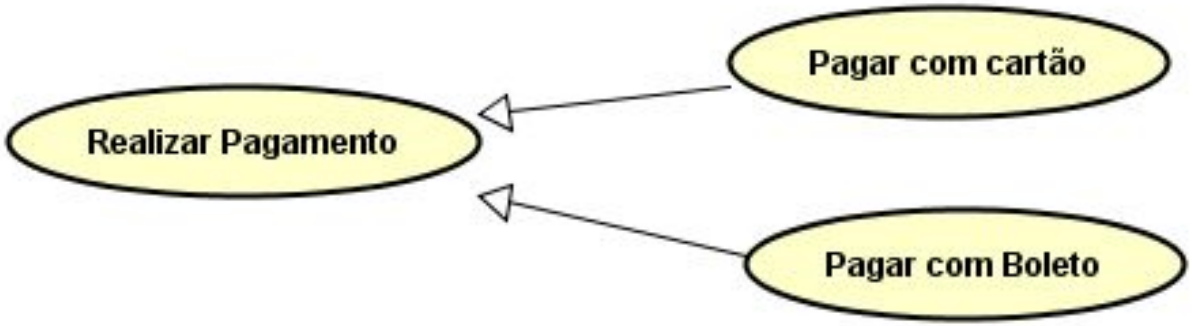


Diagrama de Casos de Uso

Exemplo

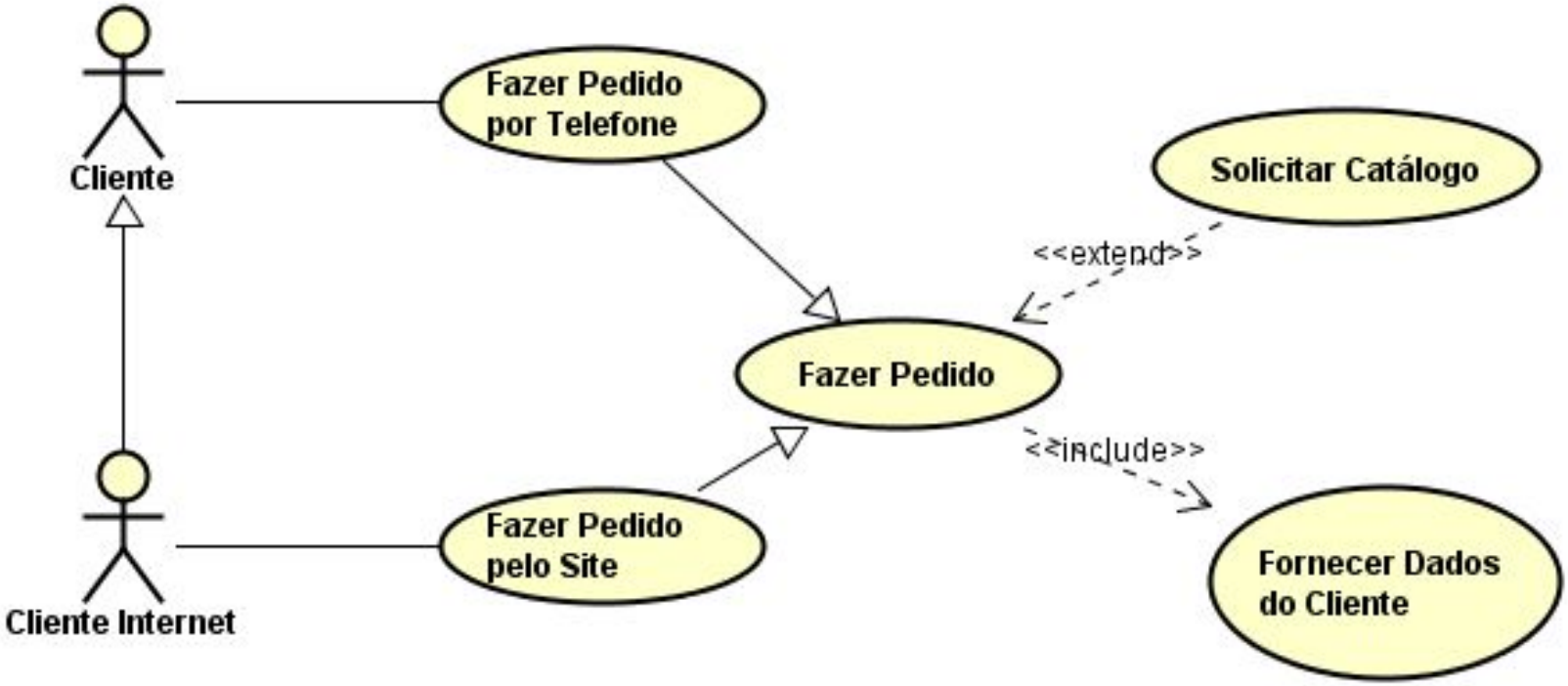
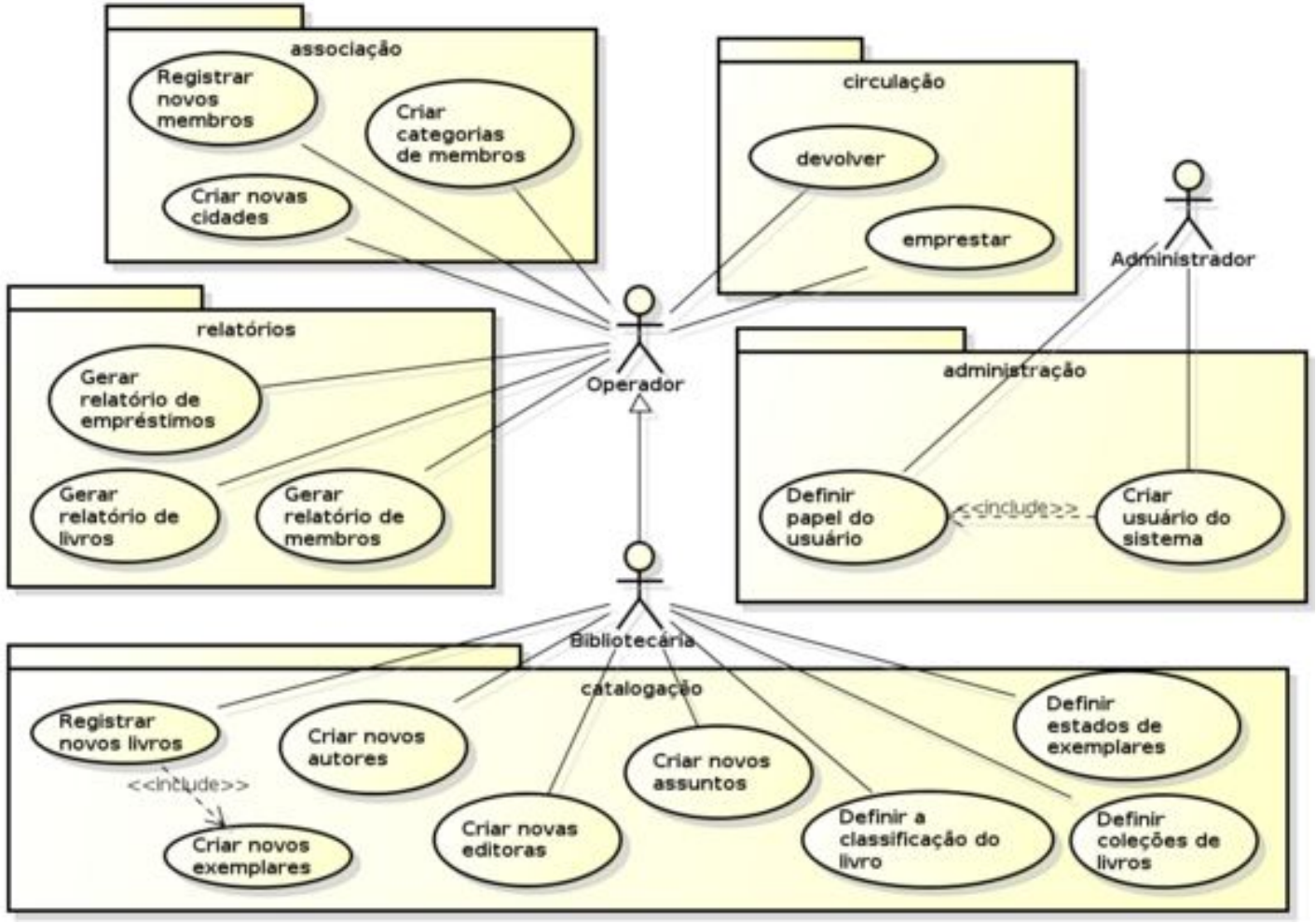


Diagrama de Casos de Uso



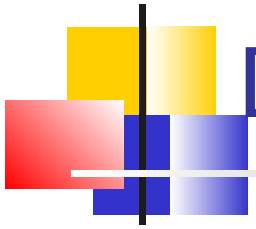
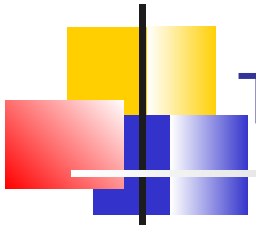


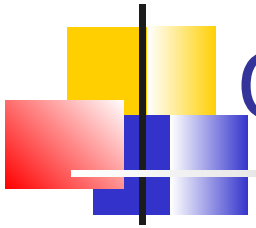
Diagrama de Classes

- Diagramas de classe são os diagramas mais comuns encontrados em modelagem de sistemas orientados a objetos.
- Um diagrama de classe mostra um conjunto de: **classes**, **interfaces** e seus **relacionamentos**.
- **Principal uso**: modelar a visão do projeto de um sistema de forma **estática**.
- São importantes para **visualização**, **especificação** e **documentação** dos modelos estruturais.



Termos e Conceitos de Diagramas de Classes (DC)

- Uma classe é uma descrição de um conjunto de objetos que compartilham: atributos, operações, relacionamentos e semântica.
- Graficamente, uma classe é desenhada como um **retângulo**.
- Uma classe em DC é composta de três partes:
  - Nome
  - Atributos
  - Métodos

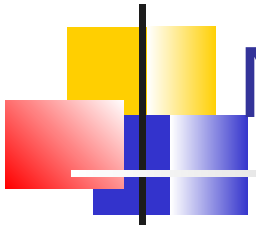


O que são: Nome, Atributo e Método

- **Nome**: Toda classe deve ter um nome que a distingue de outras.

- **Atributo**: é uma propriedade mencionada de uma classe que descreve uma variação de valores que instâncias da propriedade pode conter. A propriedade é compartilhada por todos os objetos desta classe.

- **Método**: é a implementação de um **serviço** que pode ser requerido a partir de qualquer objeto da classe para afetar seu estado.



Notações de Visibilidade em UML 2.0

- Encapsulamento:

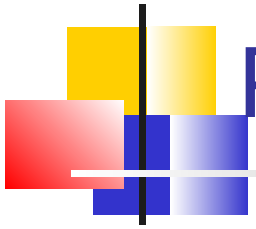
Público (+): Visível para qualquer elemento que possa ver a classe.

Protected (#): Visível a outros elementos dentro da classe e de subclasses.

Private (-): Visível a outros elementos dentro da classe.

Package (~): Visível aos elementos do pacote

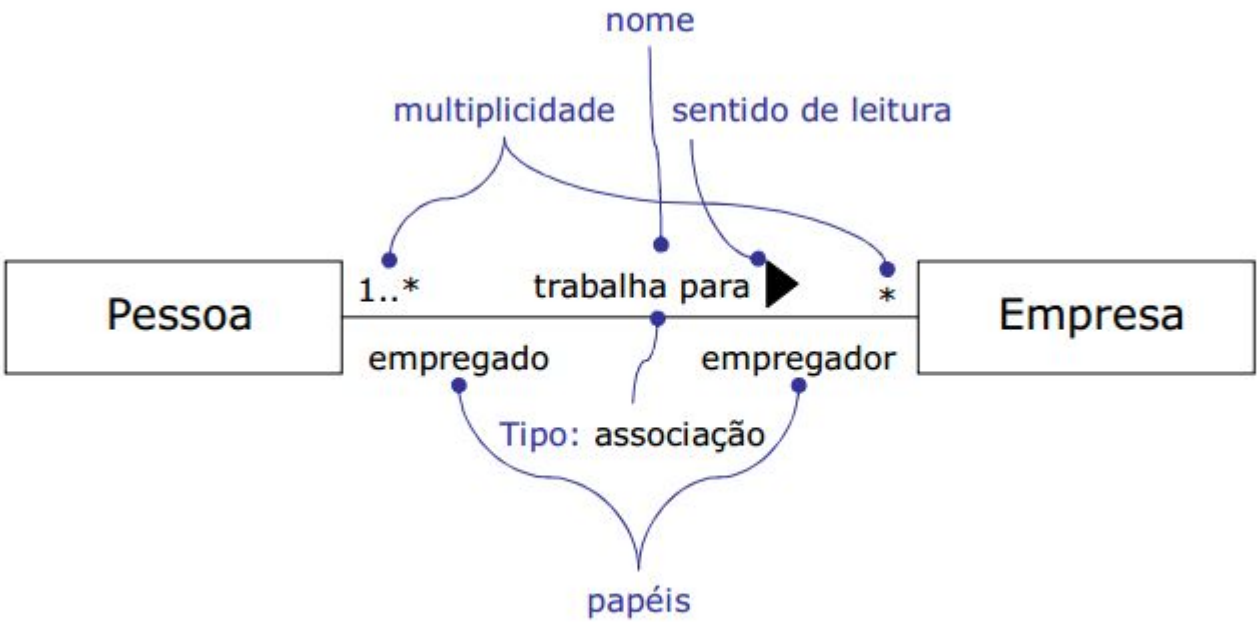
Exemplo de Classe



Relacionamento entre classes

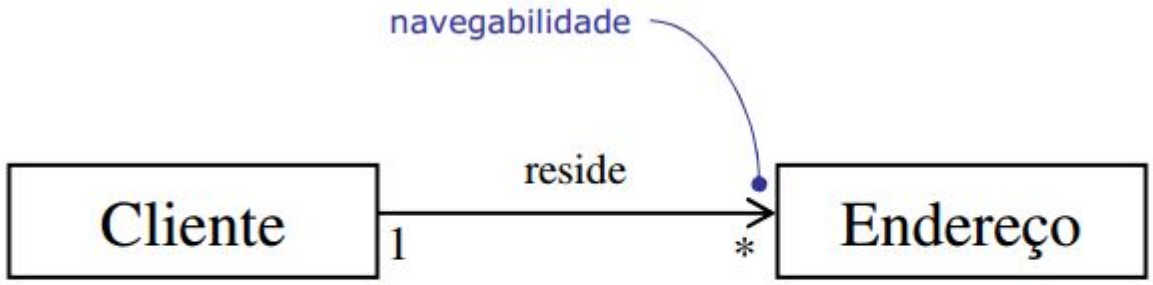
- Os relacionamentos possuem:
  - Nome: descrição dada ao relacionamento (faz, tem, possui,...)
  - Sentido de leitura
  - Navegabilidade: indicada por uma seta no fim do relacionamento
  - Multiplicidade: 0..1, 0..*, 1, 1..*, 2, 3..7
  - Tipo: associação (agregação, composição), generalização e dependência
  - Papéis: desempenhados por classes em um relacionamento

Relacionamento entre classes

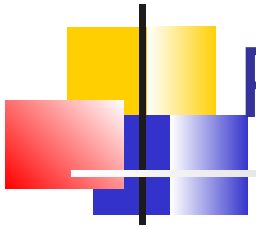


E a navegabilidade?

Relacionamento entre classes



- O cliente sabe quais são seus endereços, mas o endereço não sabe a quais clientes pertence



Relacionamento entre classes

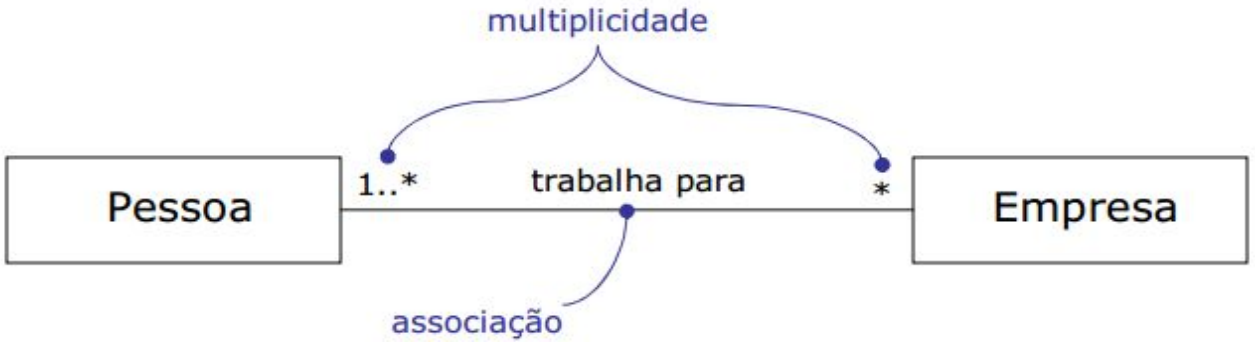
- Tipos de Relacionamento:
  - Associação
    - Agregação
    - Composição
  - Dependência
  - Generalização

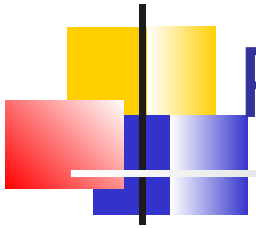
Relacionamento: Associação

- **Associação** é um relacionamento estrutural que indica que os objetos de uma classe estão vinculados a objetos de outra classe.
- Uma associação é representada por uma linha sólida conectando duas classes.

Relacionamento: Associação

- Indicadores de Multiplicidade
  - 1 Exatamente um
  - 1..* Um ou mais
  - 0..* Zero ou mais (muitos)
  - * Zero ou mais (muitos)
  - 0..1 Zero ou um
  - m..n Faixa de valores (por exemplo: 4..7)





Relacionamento: Associação

- Representação Java
  - Uma Pessoa trabalha em uma empresa

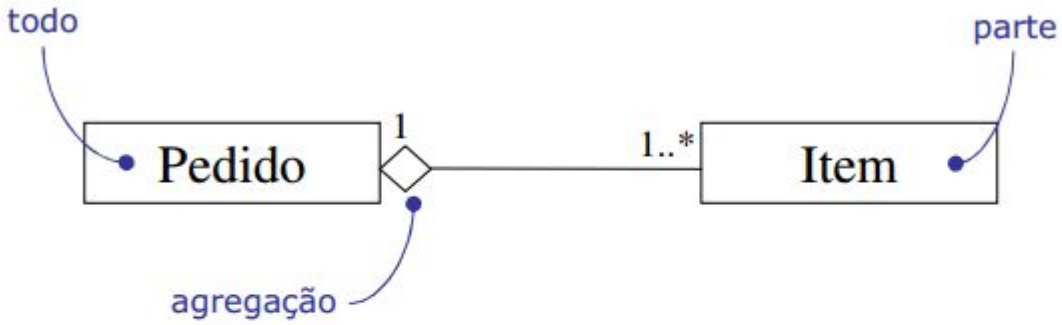
```
public class Pessoa {
    String nome;
    Empresa empresa;
}
```

  - Uma Pessoa pode trabalhar em várias empresas

```
public class Pessoa {
    String nome;
    Empresa[] empresa;
}
```

Relacionamento: Agregação

- Tipo especial de relacionamento de associação
- Usado para representar uma relação "todo - parte"



- um objeto "parte" pode fazer parte de vários objetos "todo"

Relacionamento: Agregação

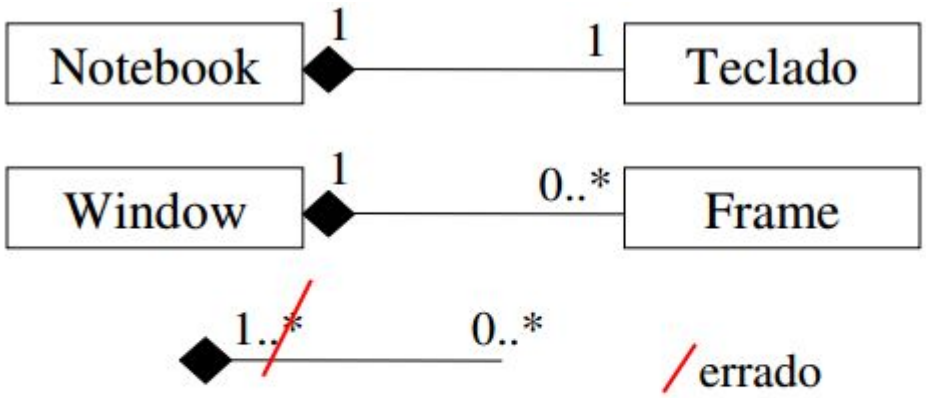
- Implementação em Java

```
public class Pedido {
   int codigo;
   ArrayList <Item> itens;

   Pedido(int codigo) {
      this.codigo = codigo;
   }
}

public class Item {
   int codigo;
   String descricao;

   Item(int codigo, String descricao) {
      this.codigo = codigo;
      this.descricao = descricao;
   }
}
```

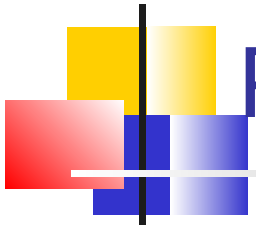
```
public class Principal {
    public static void main(String[] args) {
       Pedido pedido = new Pedido(20);

       pedido.itens.add(new Item(1,"Pão"));
       pedido.itens.add(new Item(2,"Leite"));
       pedido.itens.add(new Item(3,"Manteiga"));
    }
}
```

Relacionamento: Composição

- Tipo especial de relacionamento de associação
- Usado para representar uma relação "todo – parte" onde o objeto parte só pode pertencer a um objeto todo e tem o seu tempo de vida coincidente com o dele.



- Quando o "todo" *morre* todas as suas "partes" também *morrem*



Relacionamento: Composição

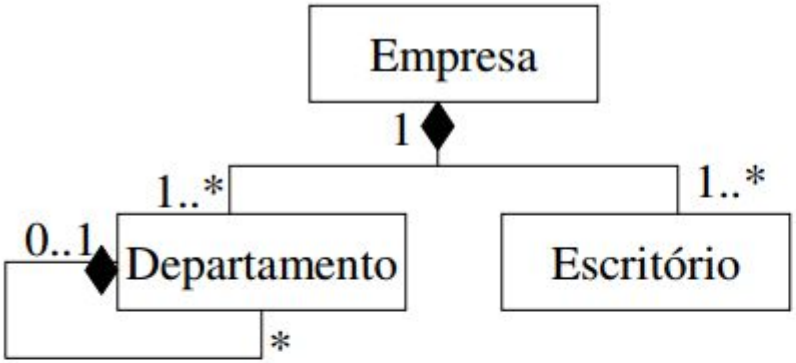
- Implementação em Java

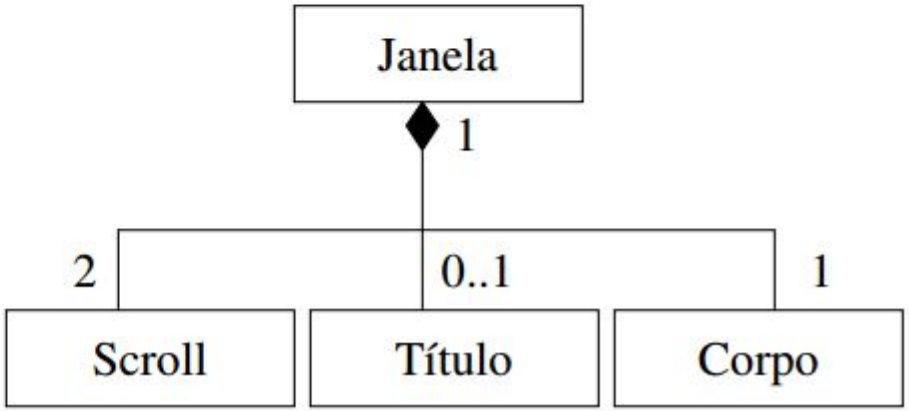
```
public class Window {
   ArrayList <Frame> frames;

   public void adicionarFrames(String descricao){
      frames.add(new Frame(descricao));
   }
}

public class Frame {
   String descricao;

   Frame(String descricao) {
      this.descricao = descricao;
   }
}
```

```
public class Principal {
    public static void main(String[] args) {
       Window window = new Window();

       window.adicionarFrames("Titulo");
       window.adicionarFrames("Menu Lateral”);
       window.adicionarFrames("Conteudo");
    }
}
```

Relacionamento: Composição

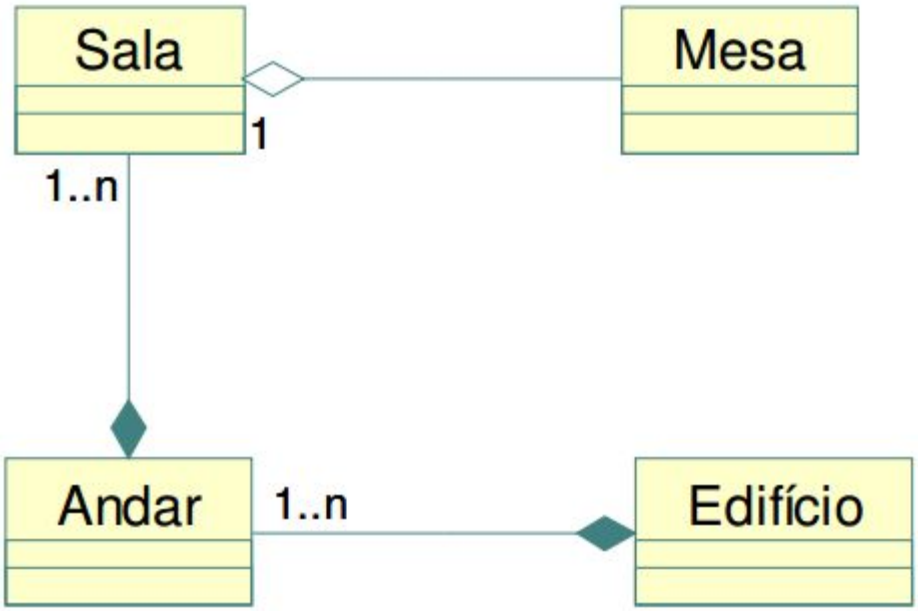
- Exemplos





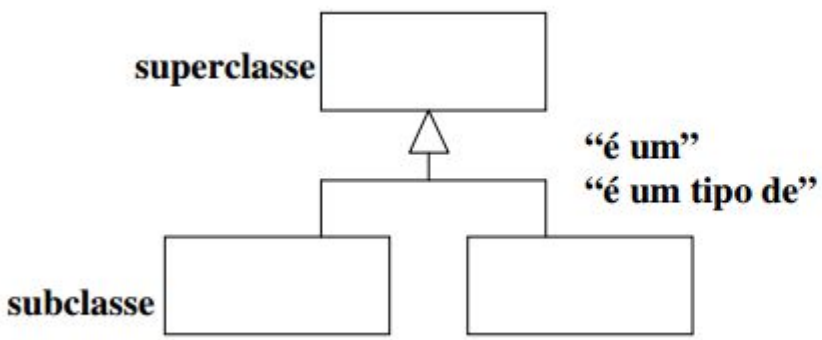
Relacionamento: Associação

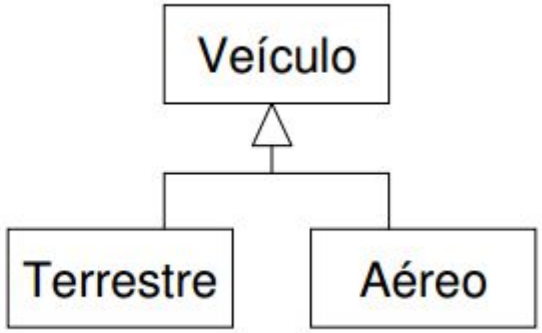
- Agregação X Composição



Relacionamento: Generalização

- É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)





Relacionamento: Dependência

- Representa que a alteração de um objeto (objeto independente) pode afetar outro objeto (objeto dependente)

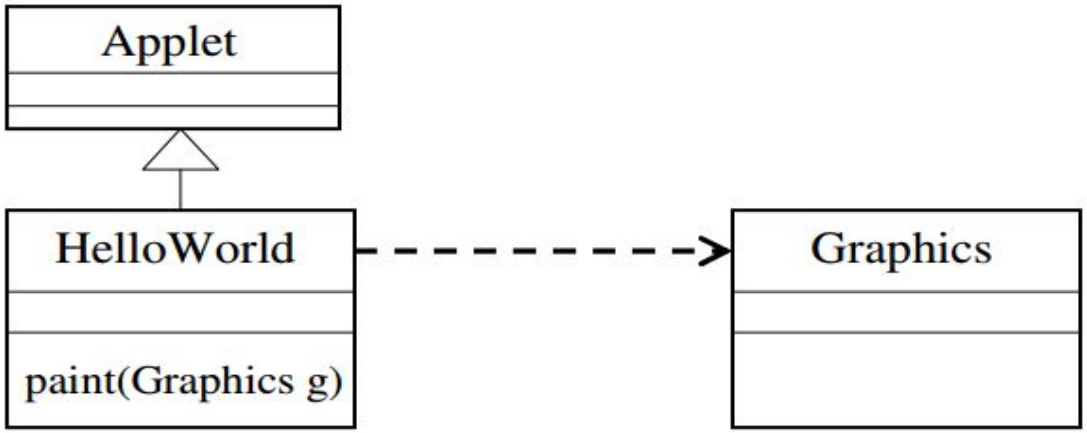
Objeto dependente Objeto independente

- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor afeta o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

Relacionamento: Dependência

- Exemplo de Implementação

```
Import java.awt.Graphics;
class HelloWorld extends java.applet.Applet
{
    public void paint (Graphics g)
        g.drawString("Hello, world!", 10, 10);
}
```



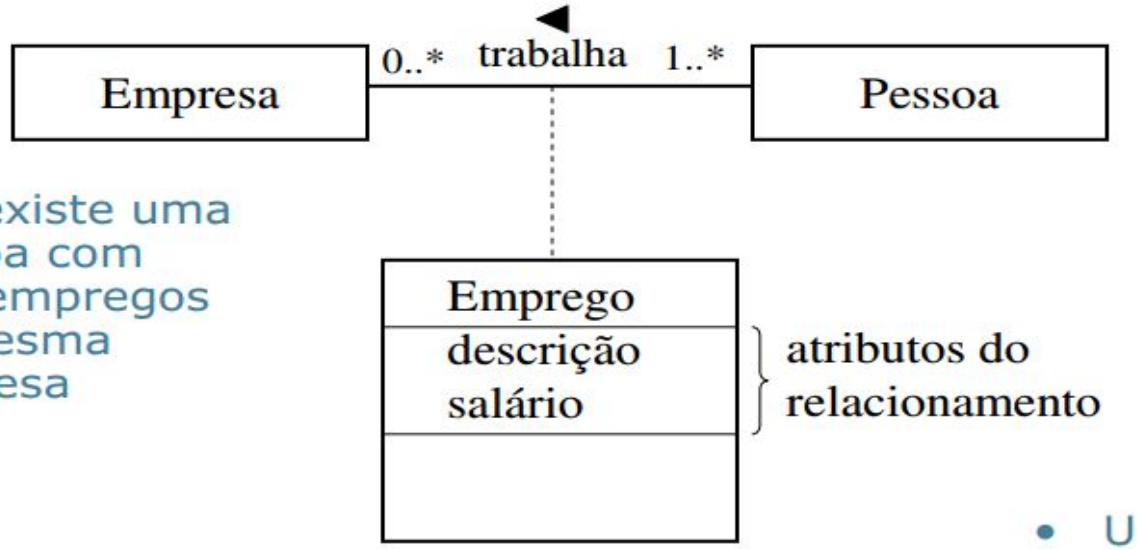
Relacionamento: Associação

- Classe de Associação
  - Usada quando uma associação entre duas classes tiver atributos da associação
  - C existe para todo relacionamento de A com B

  - C é único para o relacionamento de A e B

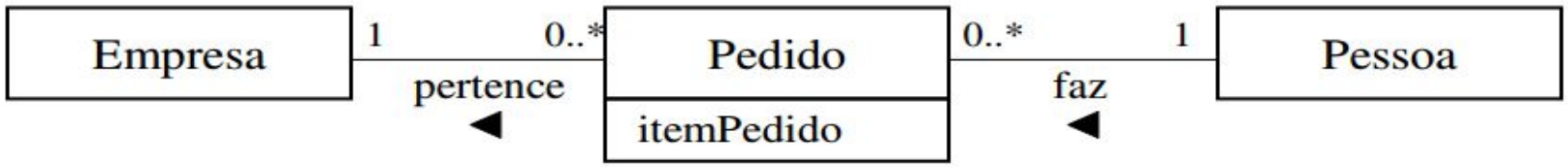
Relacionamento: Associação

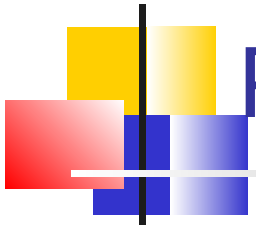
- Classe de Associação



- Não existe uma pessoa com dois empregos na mesma empresa

- Uma pessoa pode fazer mais de um pedido na mesma empresa



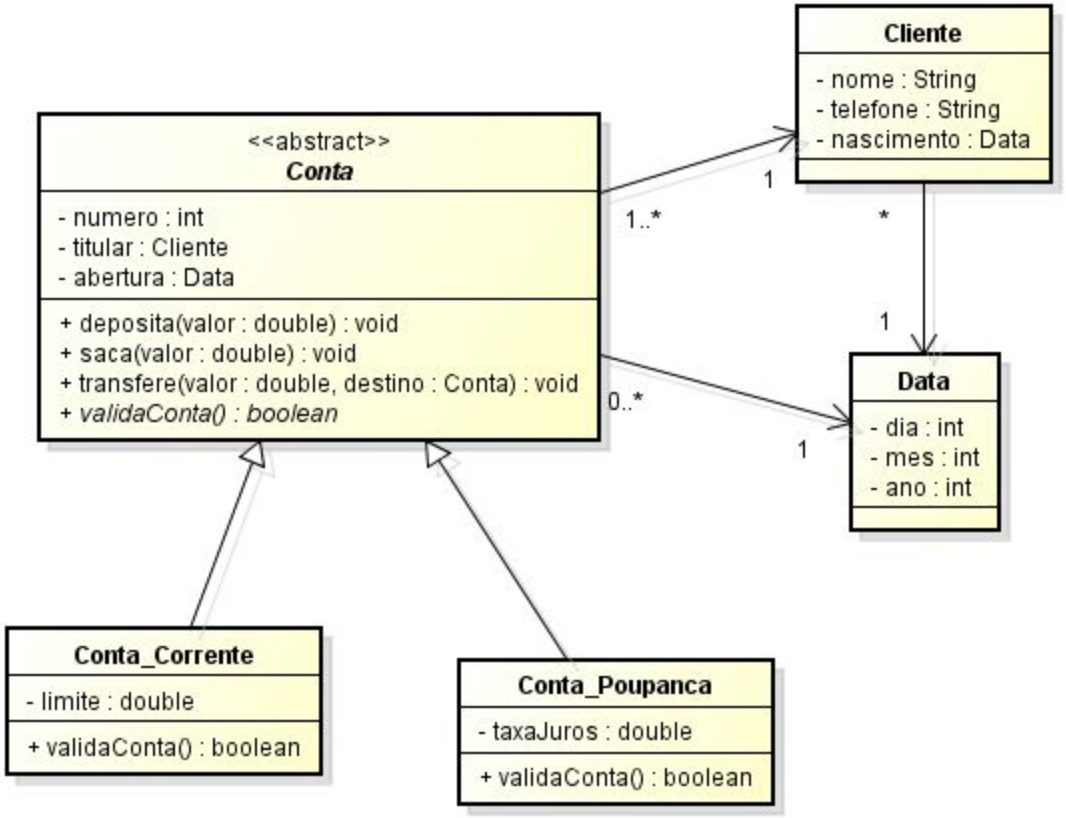


Relacionamento: Associação

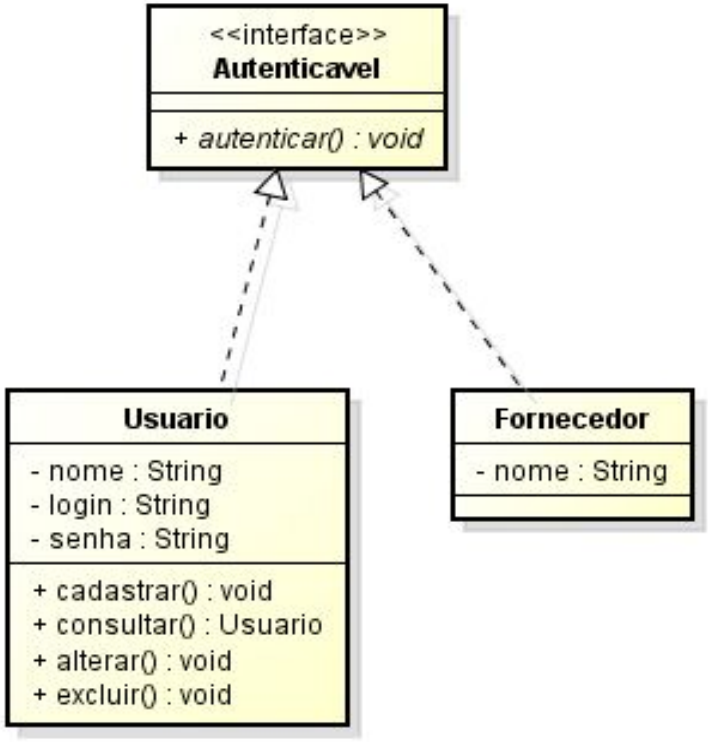
- Exemplo de Implementação

```
public class Pessoa {
    String nome;
}
public class Empresa {
    String nome;
}

public class Emprego {
    Pessoa pessoa;
    Empresa empresa;
    double salario;
    String descricao;
}
```

Classes e Métodos Abstratos - UML

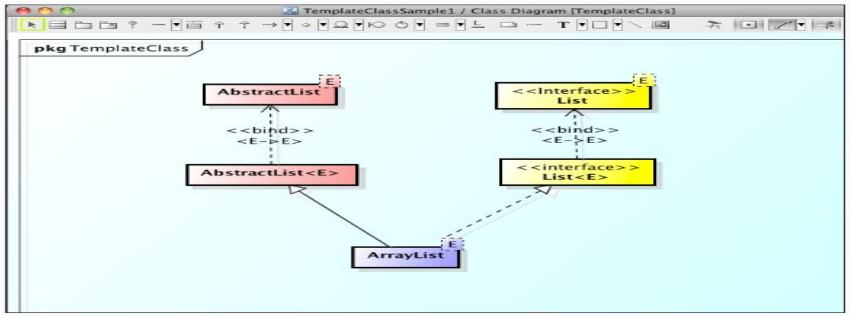


Interfaces - UML



Ferramentas para modelagem UML

- ASTAH http://astah.net/download



- Dia
- Umbrello
- ArgoUML